LAB ASSIGNMENT : 9

HTNO: 2403A52411

NAME : R.Shiavni

BATCH : 15

Assignment Title: Lab 9 – Documentation Generation: Automatic Documentation and Code Comments

Task 1: Automatic Code Commenting

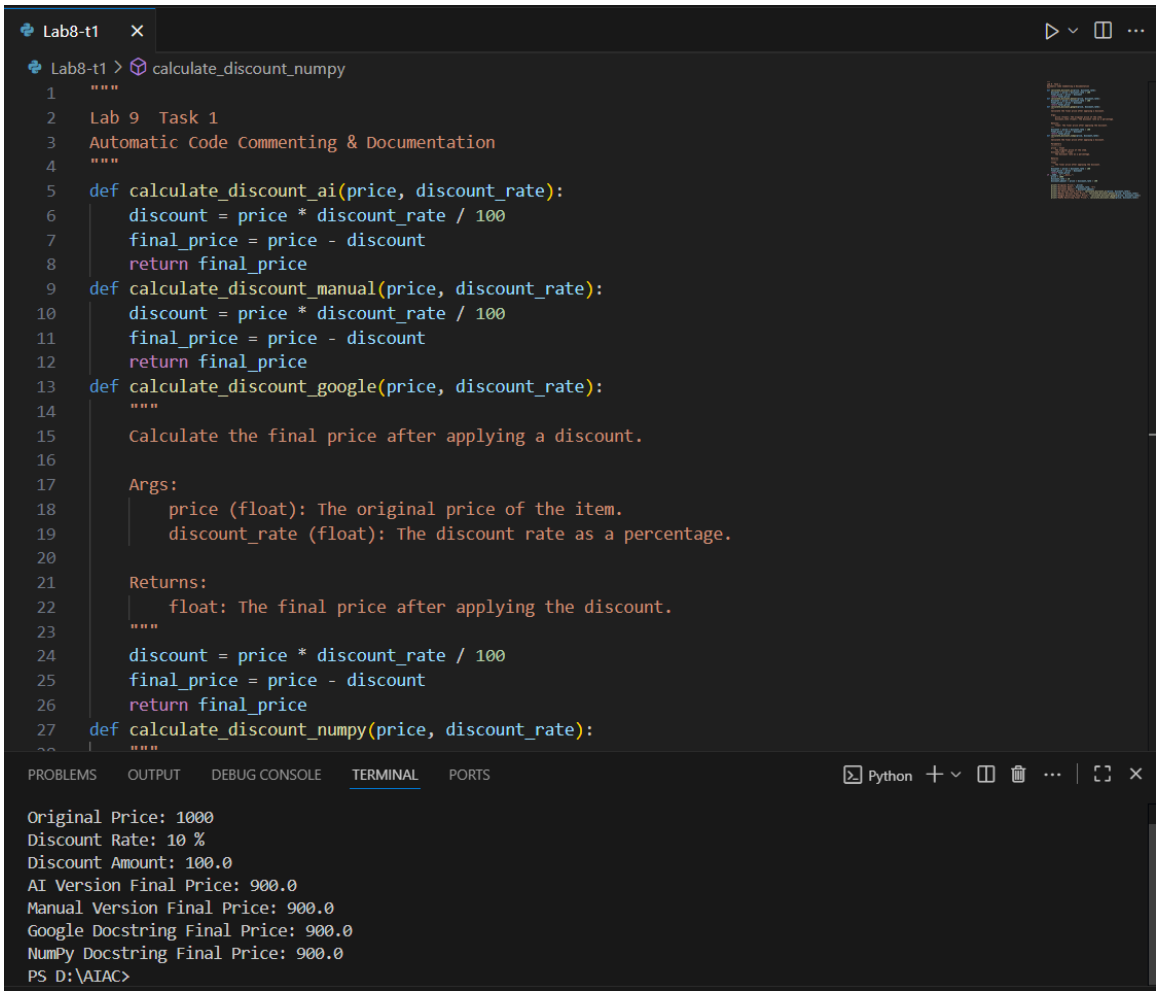
🎯 **Moto:** The main goal of Task 1 is to learn how to add inline comments and docstrings to explain code clearly, making it more understandable and maintainable. It also compares auto-generated vs manually written comments.

PROMP:

Add clear inline comments and docstrings to the AI-generated code, explaining its purpose, inputs, and logic. Make the code easier to understand and maintain. Then compare your manual comments with the AI-generated ones, highlighting differences in clarity, detail, and usefulness.

EXPLANATION: involves adding inline comments and Python docstrings to explain the logic, parameters, and return values of functions. This improves code clarity and maintainability. You'll also compare manual documentation with AI-generated comments to assess which better supports code understanding and future development in Python projects.

Code/Output:



Task 2: API Documentation Generator

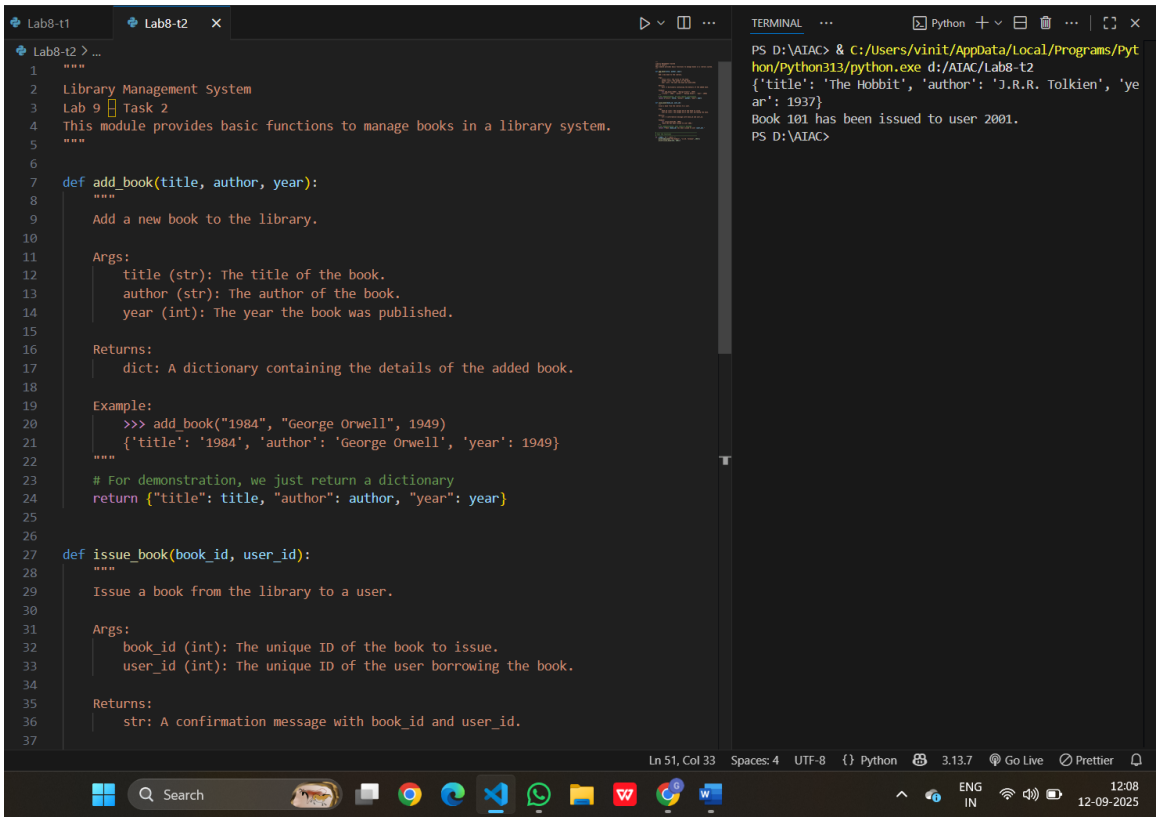
🎯 **Moto:** The main goal of Task 2 is to practice writing docstrings for functions and using auto-documentation tools like Sphinx or MkDocs to generate structured API documentation.

PROMPT: Write detailed docstrings for each function, explaining what they do, their inputs, and outputs. Then use tools like Sphinx or MkDocs to automatically generate clean, well-structured API documentation from these docstrings. Focus on making the docs clear and easy for others to use.

EXPLANATION: about writing clear, standardized docstrings for Python functions, classes, and modules, which describe their purpose, parameters, return values, and exceptions. You will then use auto-documentation tools like **Sphinx** or **MkDocs** to generate structured, professional API documentation from those docstrings. This task builds essential skills for creating maintainable and scalable documentation in real-world software projects. It emphasizes consistency, readability, and ease of navigation for developers who interact with your code. By automating

documentation from the source code, you ensure updates are easier to manage and reduce the risk of outdated or incomplete developer-facing documentation.

Code/Output:



Task 3: AI-Assisted Code Summarization

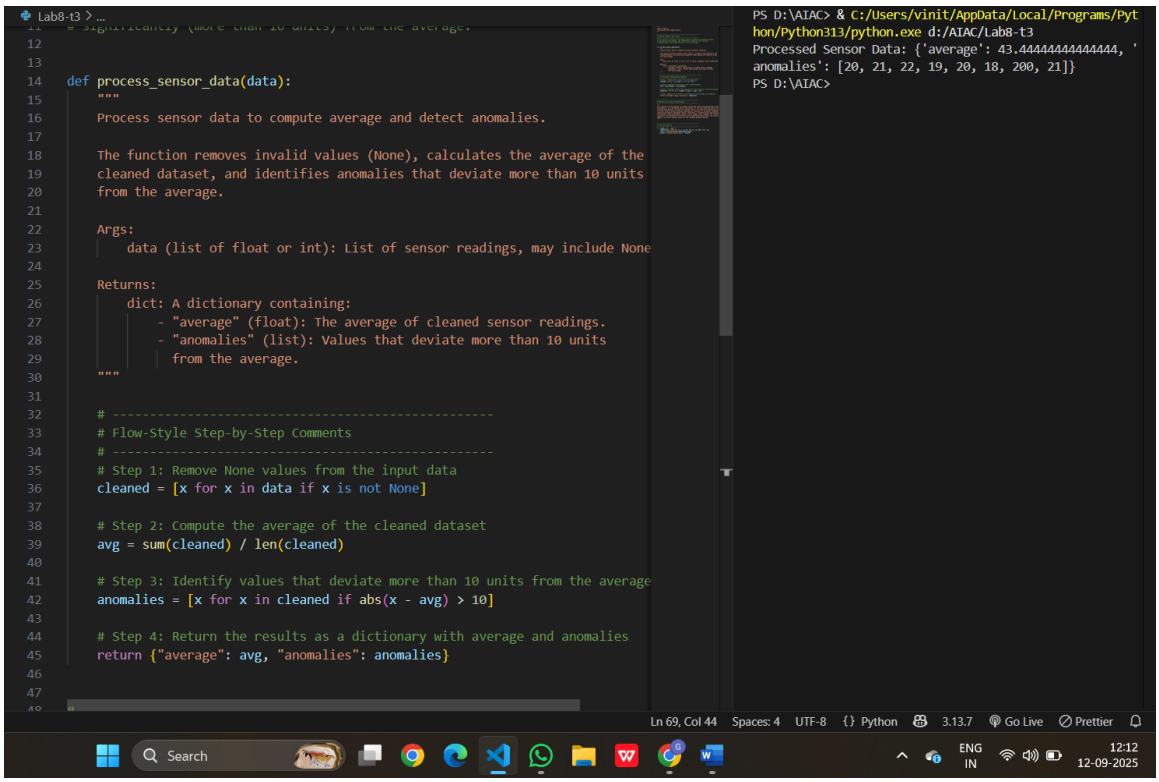
🎯 **Moto:** The goal of Task 3 is to practice summarizing functions using concise comments, step-by-step flow explanations, and documenting real-world use cases.

PROMPT: Write brief comments summarizing each function's goal. Clearly explain each step, like reading data, cleaning it, or outputting results. Include real-world use cases—for example, how the function processes user data or generates reports. Keep it practical and easy for developers to follow

EXPLANATION: designed to improve code clarity by summarizing the purpose of functions using concise comments. You will explain the logic of each function step-by-step, helping

others follow the flow of execution easily. Additionally, you'll document real-world use cases to show how the function can be applied in practical scenarios, such as handling user input, automating tasks, or processing data. This practice strengthens your ability to write readable, maintainable code and produce documentation that supports collaboration, onboarding, and debugging. The goal is to make your code self-explanatory, practical, and valuable to developers and end users alike.

Code/Output:



Task 4: Real-Time Project Documentation

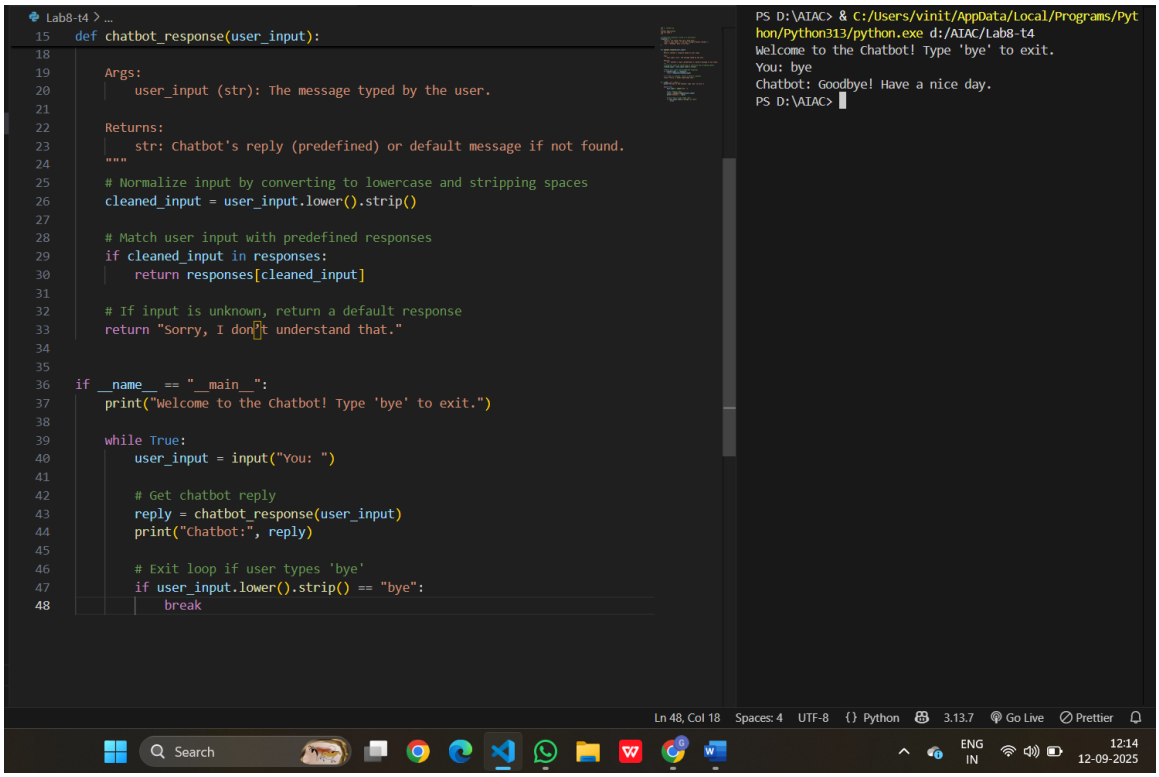
🎯 **Moto:** The main goal of Task 4 is to practice creating real-time project documentation with README, inline comments, AI-assisted guides, and reflection on automated documentation.

PROMPT: Produce real-time project documentation by creating a detailed README covering setup and usage, writing precise inline comments explaining complex code sections, and generating AI-assisted user guides. Evaluate the accuracy and completeness of automated documentation tools compared to manual documentation, emphasizing maintainability and developer onboarding.

EXPLANATION: involves creating comprehensive, real-time project documentation. This includes writing a clear README that covers project overview, installation steps, and usage

instructions. You'll also add inline comments to explain important code sections for better understanding. Using AI-assisted tools, you'll generate user guides to complement your manual documentation. Finally, you'll reflect on the quality and usefulness of automated documentation compared to your own efforts. This task emphasizes producing accessible, well-structured documentation that helps developers and users navigate the project, supports collaboration, and ensures maintainability over time.

Code/Output:



Observation

In this lab, I learned the importance of inline comments, docstrings, and documentation tools. Automated documentation keeps code easy to maintain, while manual notes provide context. Combining both approaches helps in building reliable and maintainable projects.